More Ruby

Lets look at some more basic ruby

More on Ruby Classes

- You can add another method to any class
  - any time.
  - If someone (ruby team) has written a class Fixnum
    - add a method to the class in your own code
    - class Fixnum
      - def orderOfMagnitude
        - self*10
      - end
- comments?

Now some on collections in Ruby

- Next few slides intro to ruby collections
- already seen basic strings.

String Interpolation

- can put anything into a string
  - including executable code
  - “for example #{ } “
  - anything in #{ } is executed and its result converted to a string which is inserted at that point.
  - variables, code, whatever.
  -

String manipulation

- substitutions
  - sub(<substring>, <newReplacement>)
    - replace first substring with newReplacement
  - gsub(<substring>, <newReplacement>)
    - as above, but replace all occurrences
  - regular expressions
    - /<expr>/
    - ^ = beginning of line
    - . = any character
    - $ = end of line
    - [<element1><element2> etc] any of the elements listed
    - see page 54 table 3-4 for more

Matching + split

- Might be useful for the lab
- instead of substituting
  - =~
  - “string” =~ “str”
  - “is this true” =~ /[1-4]/
- Split method
  - split(<regexp>)
    - split string into array of substrings based on regexp as separator

Arrays

- objects of course
- use [] operator as in other languages
- x = [1, 2, “three”, 4]
- zero based
- x[5] returns nothing (nil)
  - no nasty error
- add elements to end
  - <<
  -

Arrays II

- Array methods
  - pop
    - returns last element and removes it
  - length
  - join(<string>)
    - opposite of string.split
    - returns a string of all of the elements in array with <string> between each element in the resulting string.
  - inspect
    - return printable form of array.

Array Iteration.

- Iteration is object specific in ruby, arrays too
  - each method iterates over array
  - use { |<var>| <code> } notation.
  - each
    - <array>.each{<code here>}
    - non-destructively iterate over array.
  - collect
    - <array>.collect{ <code here>}
    - destructively iterate over array
      - result of code is placed in each array location.

More array methods

- methods of arrays in ruby
  - empty?
    - ? is part of the method name
    - true is array has no elements
  - include?
    - [1, “this”, 35.3].include?(“this”)
    - true if the parameter is in the array
  - named elements
    - first
    - last
      - return the first and last elements of the array respectively

Hashes

- Hashes are built in in ruby
  - hash or dictionary
  - { <key> => <value>, <key> => <value>....}
    - when writing them out.
  - eg
    - dict = {'cat'=>'feline animal', 'dog' => 'canine animal'}
    - dict.size returns 2
  - use
    - dict['cat']
      - returns 'feline animal'

Hashes II

- methods on hashes
  - each # functions like each on arrays
    - passes two variables to the code block though.
  - keys
    - retrieves and array of the keys
  - delete(<key>)
    - deletes the <key>=><value> pair from the hash

Ranges

- Ruby has a class with a notion of a range
  - numeric, alphabetic and so on
  - eg
    - ('A'..'Z')
    - (1..100)
  - methods (many similar to arrays and all collections)
  - to_a (converts range to array with all elements filled out.
  - include?

Opening a file

- Two ways
  - File.open(<filename>)<code block>
    - opens <file name> and executes code block then closes file.
    - seems to be most common for reading
    - use each method to do something with each line.
  - File.new(<filename>, <mode>)
    - returns a file object, filename and mode are both strings
    - mode is standard modes r, w, a, rw, etc
  - need to call close on the resulting object when done in this scenario.

Reading Assignment

- Read chapters 4 and 5 in the ruby book.
- again cheerleading warning in chapter 5
- strong strong STRONG encouragement: read Chapter 4 before beginning the lab.